Freshworks Assignment Output Samples

Hello dear,
This file contains pictures of some basic test cases added to the project. You can go through it and can easily understand How it is working.

Let jump into the main content
Here I have shown you outputs of accessing database using two methods:

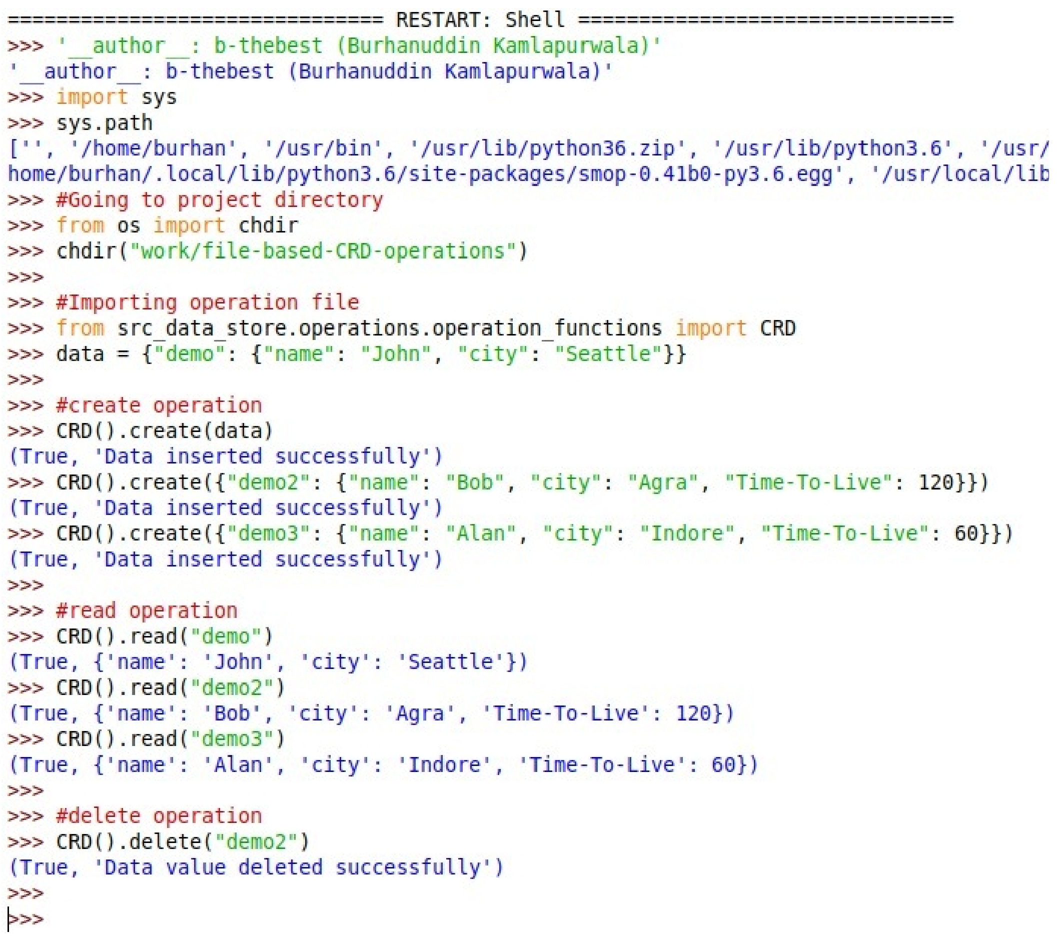
- Through RESTful interface (API).
- By importing module to other module and use functions.

I have implemented Create, Read, Delete (CRD) operation using file based database with all possible errors in it.

NOTE: If any picture is not visible, it can be found at github repository

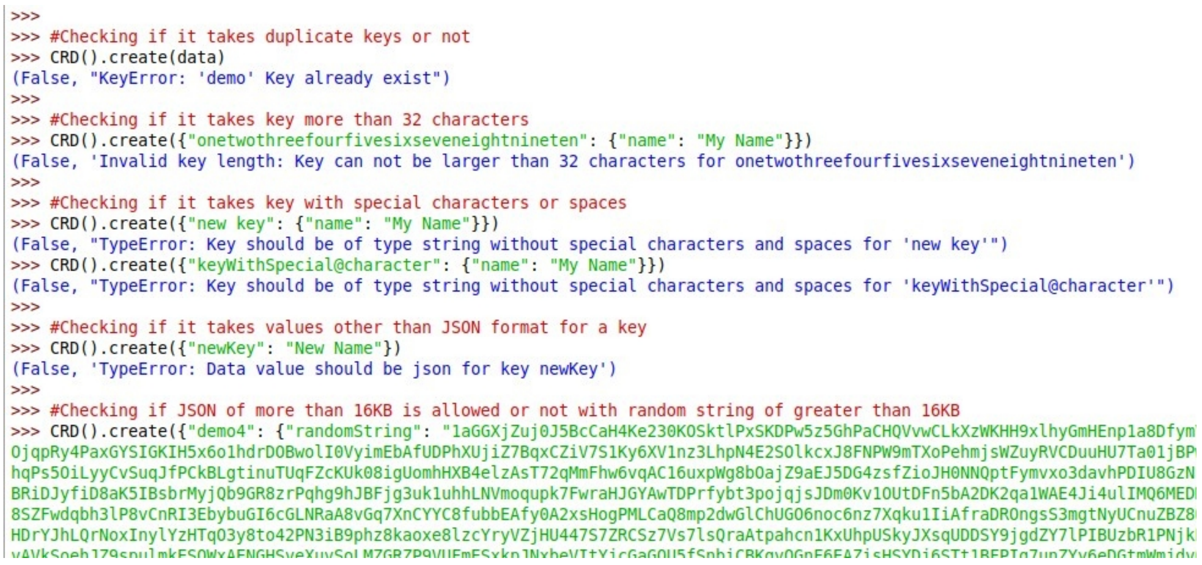
Using module in another module

1. In the below image you can see how basic operations are working after importing the module.

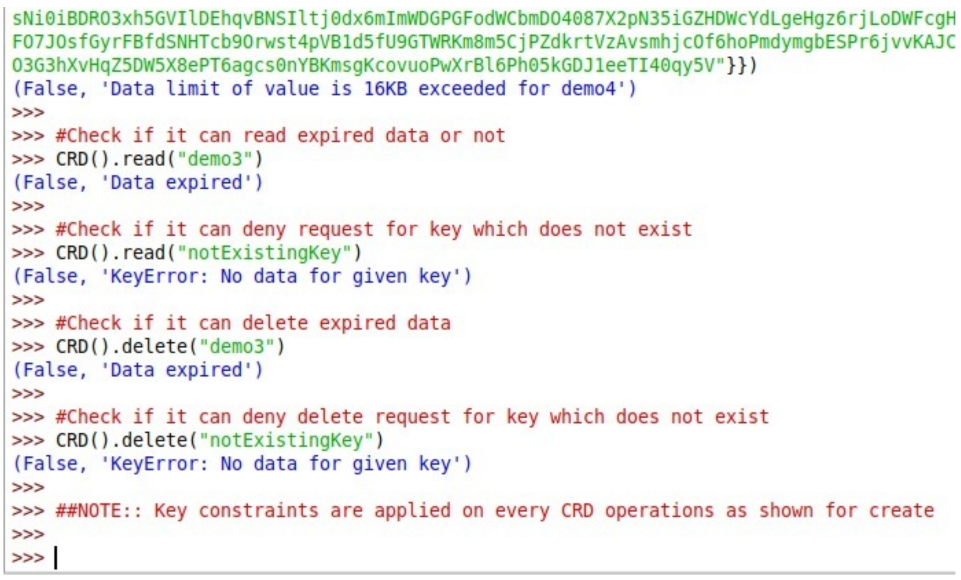


```
=============================== RESTART: Shell ===============================
>>> '__author__: b-thebest (Burhanuddin Kamlapurwala)'
'__author__: b-thebest (Burhanuddin Kamlapurwala)'
>>> import sys
>>> sys.path
['', '/home/burhan', '/usr/bin', '/usr/lib/python36.zip', '/usr/lib/python3.6', '/usr/
home/burhan/.local/lib/python3.6/site-packages/smop-0.41b0-py3.6.egg', '/usr/local/lib
>>> #Going to project directory
>>> from os import chdir
>>> chdir("work/file-based-CRD-operations")
>>>
>>> #Importing operation file
>>> from src_data_store.operations.operation_functions import CRD
>>> data = {"demo": {"name": "John", "city": "Seattle"}}
>>>
>>> #create operation
>>> CRD().create(data)
(True, 'Data inserted successfully')
>>> CRD().create({"demo2": {"name": "Bob", "city": "Agra", "Time-To-Live": 120}})
(True, 'Data inserted successfully')
>>> CRD().create({"demo3": {"name": "Alan", "city": "Indore", "Time-To-Live": 60}})
(True, 'Data inserted successfully')
>>>
>>> #read operation
>>> CRD().read("demo")
(True, {'name': 'John', 'city': 'Seattle'})
>>> CRD().read("demo2")
(True, {'name': 'Bob', 'city': 'Agra', 'Time-To-Live': 120})
>>> CRD().read("demo3")
(True, {'name': 'Alan', 'city': 'Indore', 'Time-To-Live': 60})
>>>
>>> #delete operation
>>> CRD().delete("demo2")
(True, 'Data value deleted successfully')
>>>
>>>
```

2. In the below file attached we will be able to see different types of constraint applied while using CRD functions.



```
>>>
>>> #Checking if it takes duplicate keys or not
>>> CRD().create(data)
(False, "KeyError: 'demo' Key already exist")
>>>
>>> #Checking if it takes key more than 32 characters
>>> CRD().create({"onetwothreefourfivesixseveneightnineten": {"name": "My Name"}})
(False, 'Invalid key length: Key can not be larger than 32 characters for onetwothreefourfivesixseveneightnineten')
>>>
>>> #Checking if it takes key with special characters or spaces
>>> CRD().create({"new key": {"name": "My Name"}})
(False, "TypeError: Key should be of type string without special characters and spaces for 'new key'")
>>> CRD().create({"keyWithSpecial@character": {"name": "My Name"}})
(False, "TypeError: Key should be of type string without special characters and spaces for 'keyWithSpecial@character'")
>>>
>>> #Checking if it takes values other than JSON format for a key
>>> CRD().create({"newKey": "New Name"})
(False, 'TypeError: Data value should be json for key newKey')
>>>
>>> #Checking if JSON of more than 16KB is allowed or not with random string of greater than 16KB
>>> CRD().create({"demo4": {"randomString": "1aGGXjZuj0J5BcCaH4Ke230KOSktlPxSKDPw5z5GhPaCHQVvwCLkXzWKHH9xlhyGmHEnp1a8Dfym
OjqpRy4PaxGYSIGKIH5x6o1hdrDOBwolI0VyimEbAfUDPhXUjiZ7BqxCZiV7S1Ky6XV1nz3LhpN4E2SOlkcxJ8FNPW9mTXoPehmjsWZuyRVCDuuHU7Ta01jBP
hqPs5OiLyyCvSuqJfPCkBLgtinuTUqFZcKUk08igUomhHXB4elzAsT72qMmFhw6vqAC16uxpWg8bOajZ9aEJ5DG4zsfZioJH0NNQptFymvxo3davhPDIU8GzN
BRiDJyfiD8aK5IBsbrMyjQb9GR8zrPqhg9hJBFjg3uk1uhhLNVmoqupk7FwraHJGYAwTDPrfybt3pojqjsJDm0Kv1OUtDFn5bA2DK2qa1WAE4Ji4ulIMQ6MED
8SZFwdqbh3lP8vCnRI3EbybuGI6cGLNRaA8vGq7XnCYYC8fubbEAfy0A2xsHogPMLCaQ8mp2dwGlChUGO6noc6nz7Xqku1IiAfraDROngsS3mgtNyUCnuZBZ8
HDrYJhLQrNoxInylYzHTqO3y8to42PN3iB9phz8kaoxe8lzcYryVZjHU447S7ZRCSz7Vs7lsQraAtpahcn1KxUhpUSkyJXsqUDDSY9jgdZY7lPIBUzbR1PNjk
vAVkSoehJZ9spulmkFSOWxAENGHSveXuvSoLMZGRZP9VUEmESxknJNxbeVItYicGaGOU5fSnbiCBKqvOGnE6EAZisHSYDj6STt1BEPIq7unZYv6eDGtmWmidv
```

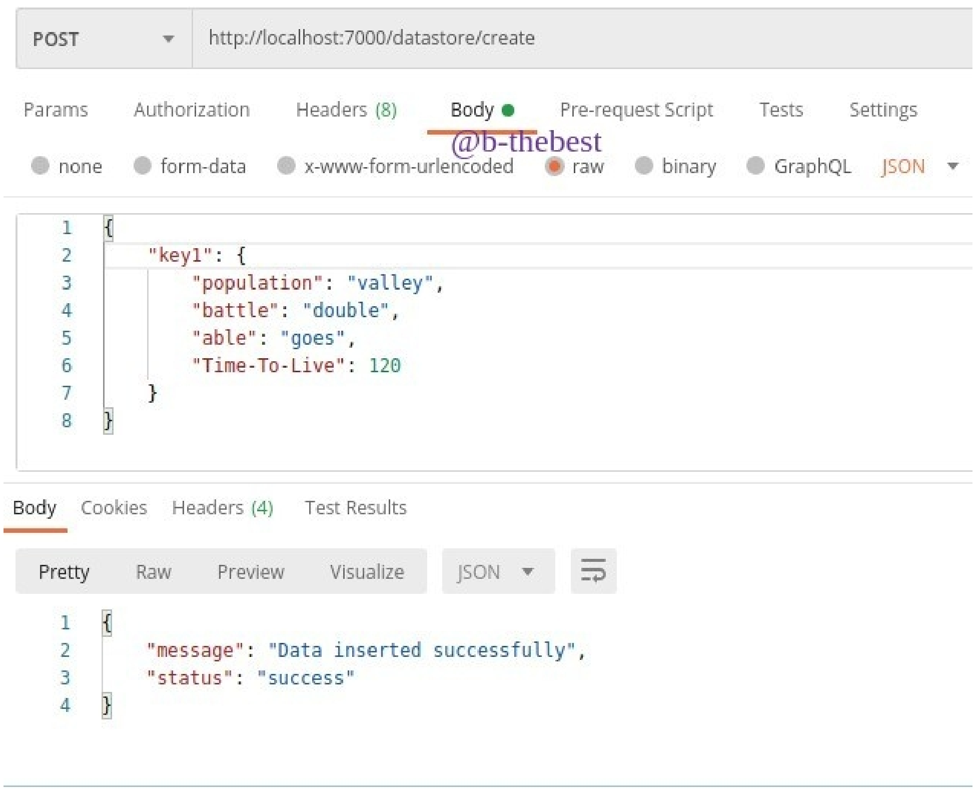


```
sNi0iBDRO3xh5GVIlDEhqvBNSIltj0dx6mImWDGPGFodWCbmDO4087X2pN35iGZHDWcYdLgeHgz6rjLoDWFcgH
FO7JOsfGyrFBfdSNHTcb9Orwst4pVB1d5fU9GTWRKm8m5CjPZdkrtVzAvsmhjcOf6hoPmdymgbESPr6jvvKAJC
O3G3hXvHqZ5DW5X8ePT6agcs0nYBKmsgKcovuoPwXrBl6Ph05kGDJ1eeTI40qy5V"}})
(False, 'Data limit of value is 16KB exceeded for demo4')
>>>
>>> #Check if it can read expired data or not
>>> CRD().read("demo3")
(False, 'Data expired')
>>>
>>> #Check if it can deny request for key which does not exist
>>> CRD().read("notExistingKey")
(False, 'KeyError: No data for given key')
>>>
>>> #Check if it can delete expired data
>>> CRD().delete("demo3")
(False, 'Data expired')
>>>
>>> #Check if it can deny delete request for key which does not exist
>>> CRD().delete("notExistingKey")
(False, 'KeyError: No data for given key')
>>>
>>> ##NOTE:: Key constraints are applied on every CRD operations as shown for create
>>>
>>> |
```

RESTful Interface

Here we go to see sample of REST API results.

1. Create some record

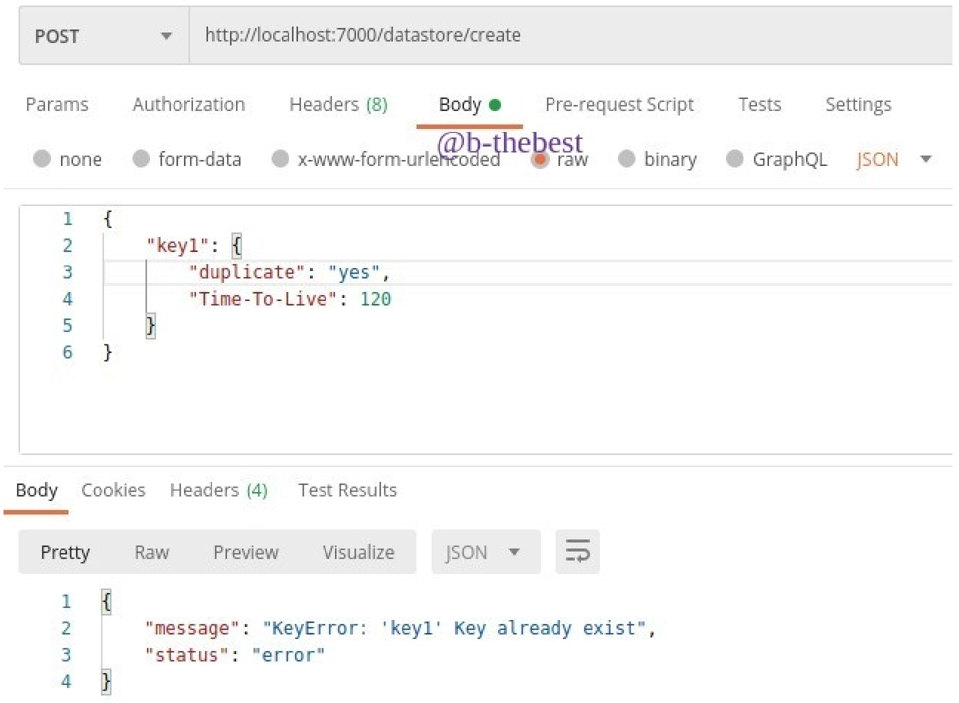


http://localhost:7000/datastore/create

```
{
    "key1": {
        "population": "valley",
        "battle": "double",
        "able": "goes",
        "Time-To-Live": 120
    }
}
```

```
{
    "message": "Data inserted successfully",
    "status": "success"
}
```

2. Trying to create duplicate key.

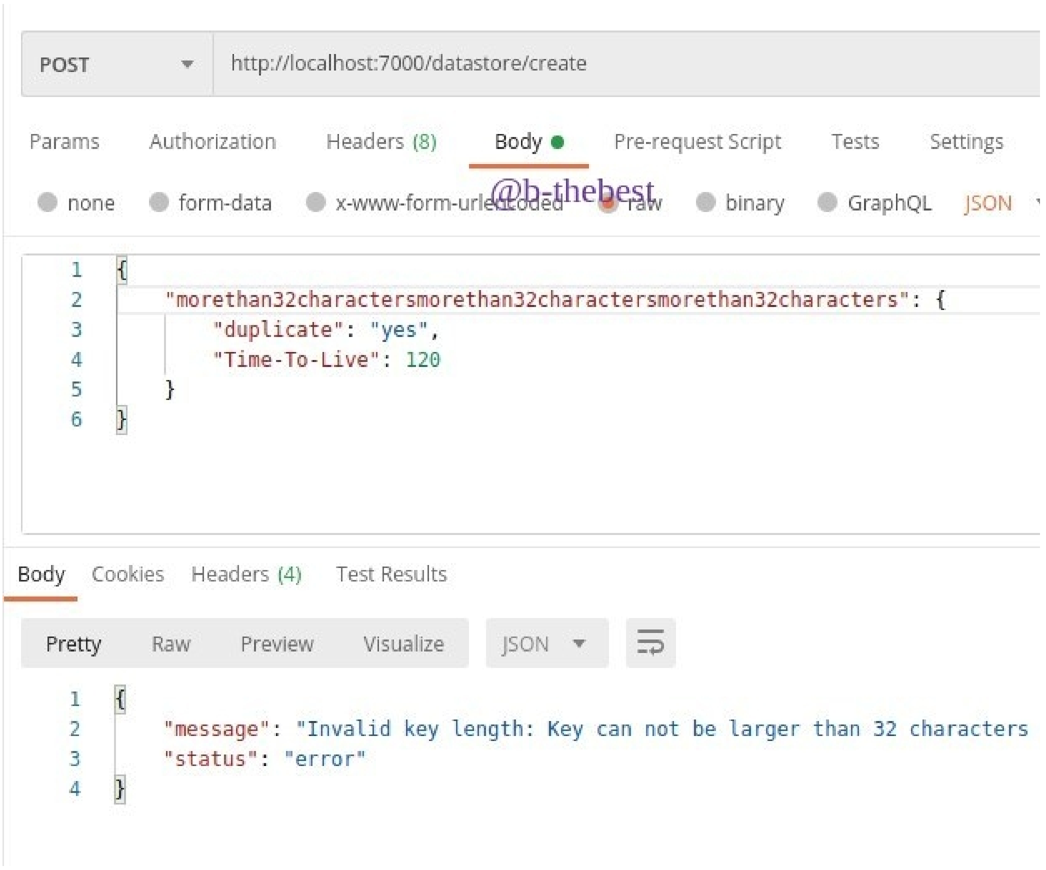


http://localhost:7000/datastore/create

```
{
    "key1": {
        "duplicate": "yes",
        "Time-To-Live": 120
    }
}
```

```
{
    "message": "KeyError: 'key1' Key already exist",
    "status": "error"
}
```

3. Trying to create record with key length more than 32 characters



POST http://localhost:7000/datastore/create

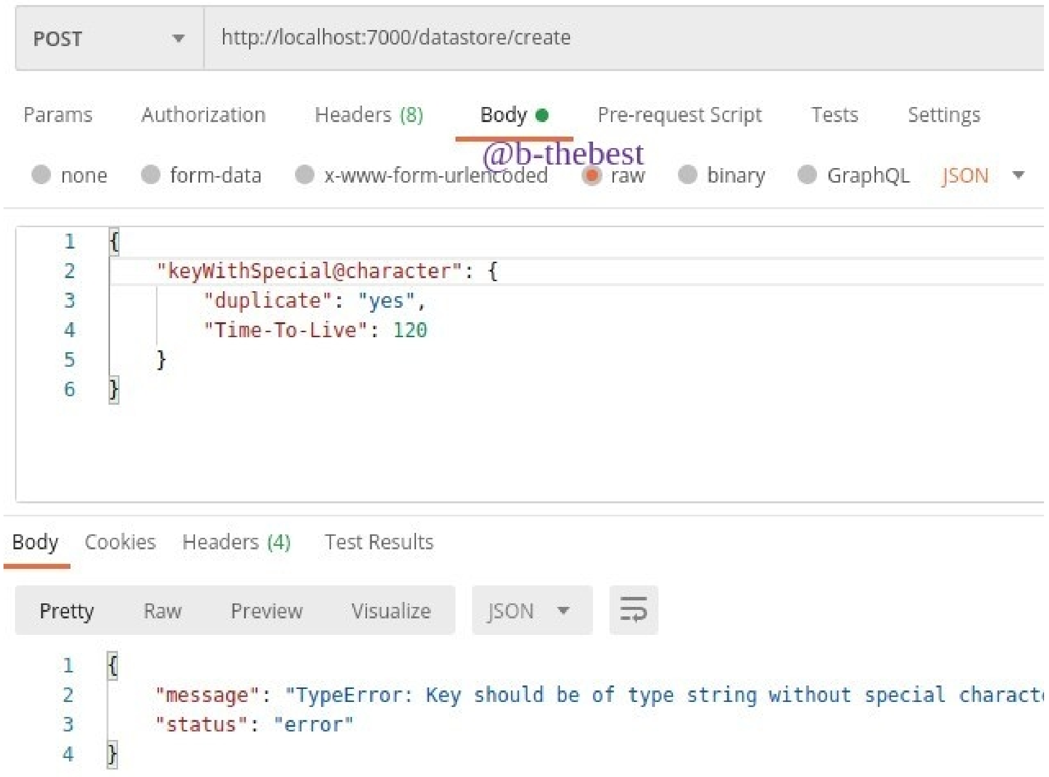
Body: raw JSON

```
{
    "morethan32charactersmorethan32charactersmorethan32characters": {
        "duplicate": "yes",
        "Time-To-Live": 120
    }
}
```

Response Body (JSON):

```
{
    "message": "Invalid key length: Key can not be larger than 32 characters
    "status": "error"
}
```

4. Key containing special character should not be allowed



POST http://localhost:7000/datastore/create

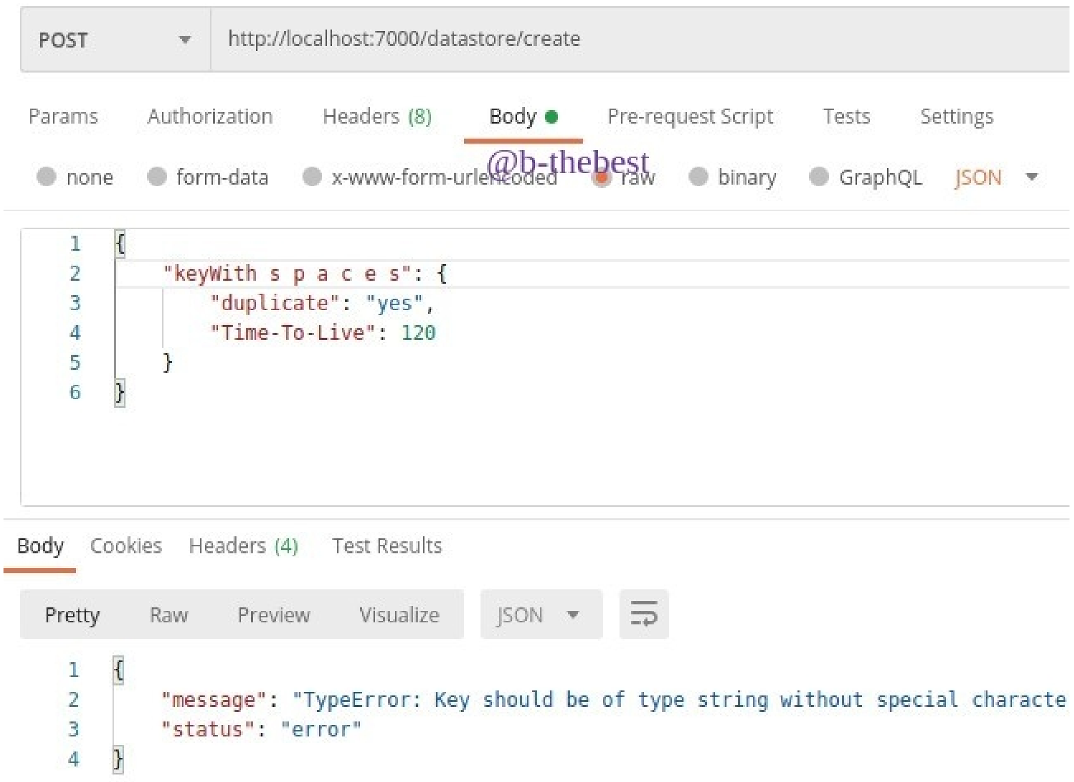
Body: raw JSON

```
{
    "keyWithSpecial@character": {
        "duplicate": "yes",
        "Time-To-Live": 120
    }
}
```

Response Body (JSON):

```
{
    "message": "TypeError: Key should be of type string without special charact
    "status": "error"
}
```

5. Key containing spaces should not be allowed

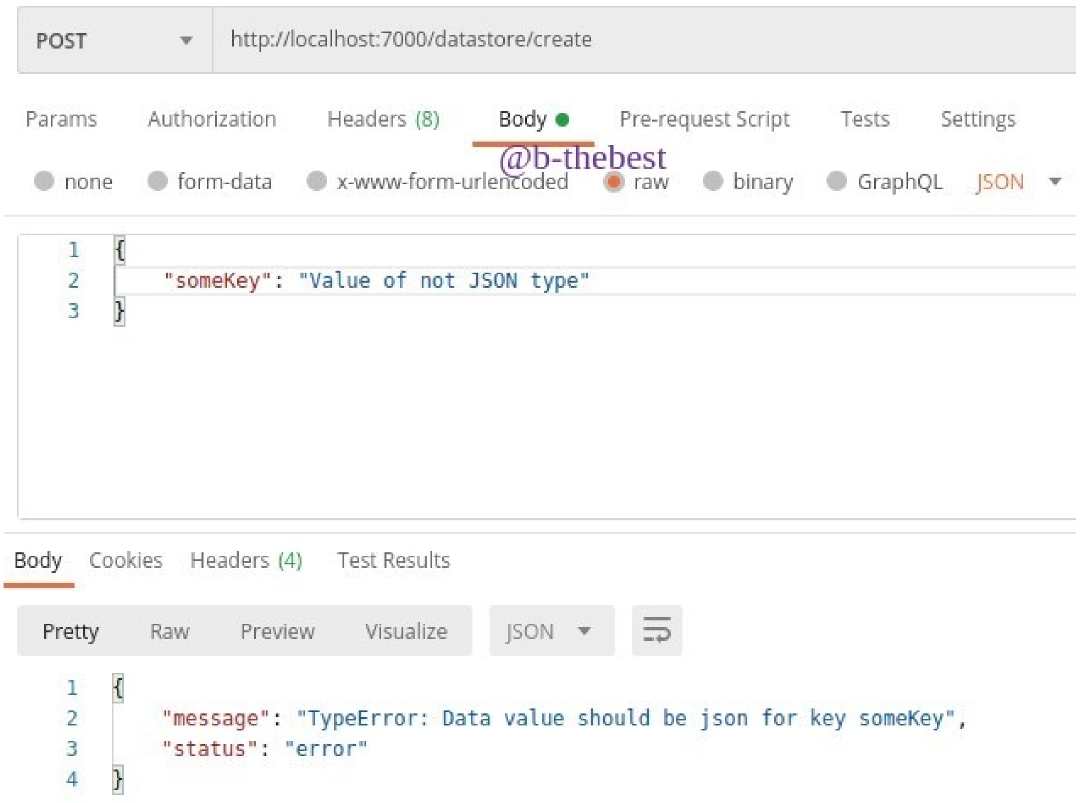


```
http://localhost:7000/datastore/create
```

```
{
    "keyWith s p a c e s": {
        "duplicate": "yes",
        "Time-To-Live": 120
    }
}
```

```
{
    "message": "TypeError: Key should be of type string without special characte
    "status": "error"
}
```

6. Value should always be JSON type

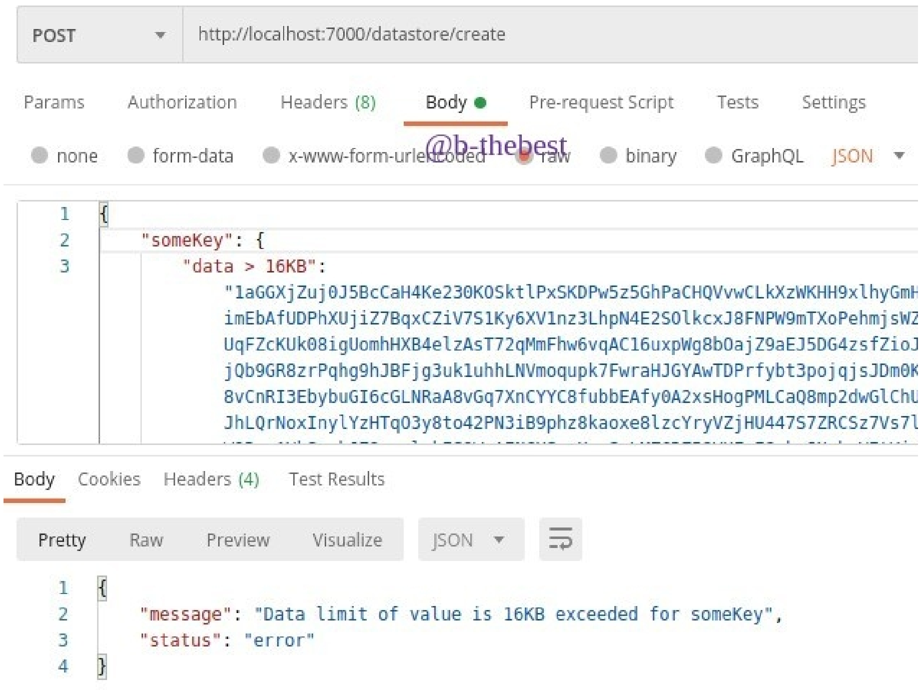


```
http://localhost:7000/datastore/create
```

```
{
    "someKey": "Value of not JSON type"
}
```

```
{
    "message": "TypeError: Data value should be json for key someKey",
    "status": "error"
}
```

7. JSON size should not exceed 16KB

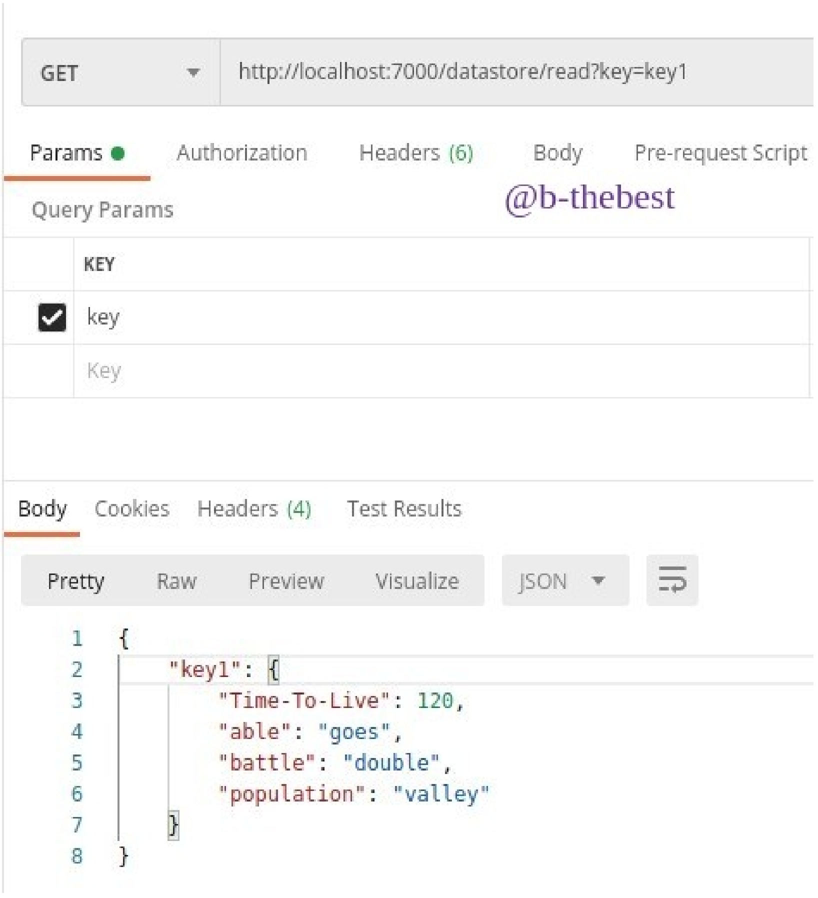


http://localhost:7000/datastore/create

```
{
    "someKey": {
        "data > 16KB":
            "1aGGXjZuj0J5BcCaH4Ke230KOSktlPxSKDPw5z5GhPaCHQVvwCLkXzWKHH9xlhyGmH
            imEbAfUDPhXUjiZ7BqxCZiV7S1Ky6XV1nz3LhpN4E2SOlkcxJ8FNPW9mTXoPehmjsWZ
            UqFZcKUk08igUomhHXB4elzAsT72qMmFhw6vqAC16uxpWg8bOajZ9aEJ5DG4zsfZioJ
            jQb9GR8zrPqhg9hJBFjg3uk1uhhLNVmoqupk7FwraHJGYAwTDPrfybt3pojqjsJDm0K
            8vCnRI3EbybuGI6cGLNRaA8vGq7XnCYYC8fubbEAfy0A2xsHogPMLCaQ8mp2dwGlChU
            JhLQrNoxInylYzHTqO3y8to42PN3iB9phz8kaoxe8lzcYryVZjHU447S7ZRCSz7Vs7l
```

```
{
    "message": "Data limit of value is 16KB exceeded for someKey",
    "status": "error"
}
```

8. Reading data value present in database

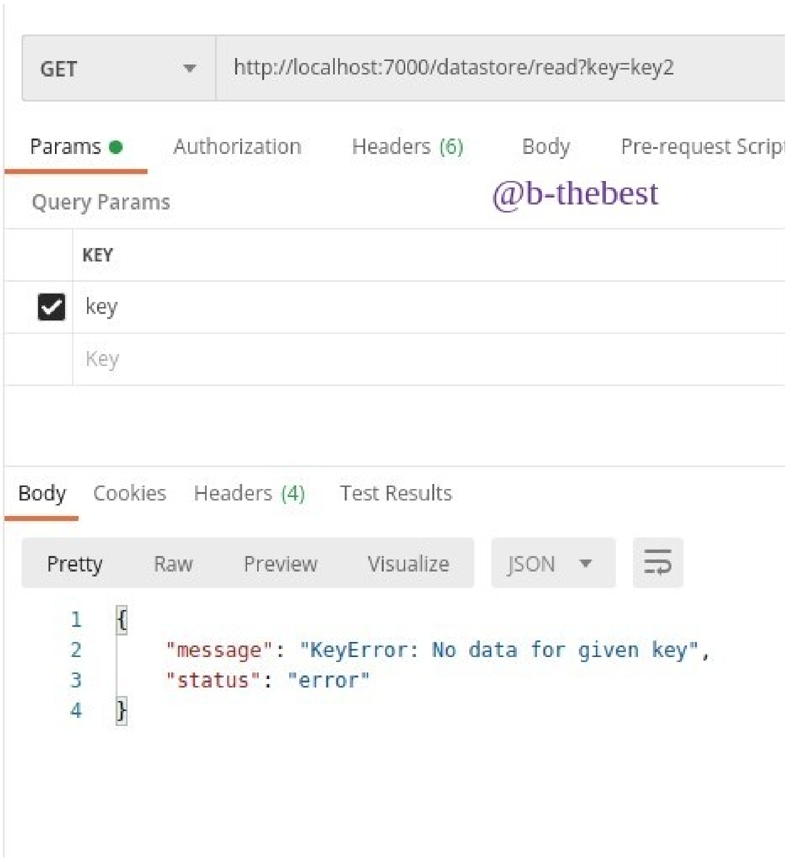


http://localhost:7000/datastore/read?key=key1

	KEY
✓	key

```
{
    "key1": {
        "Time-To-Live": 120,
        "able": "goes",
        "battle": "double",
        "population": "valley"
    }
}
```

9. Reading for key not present in database



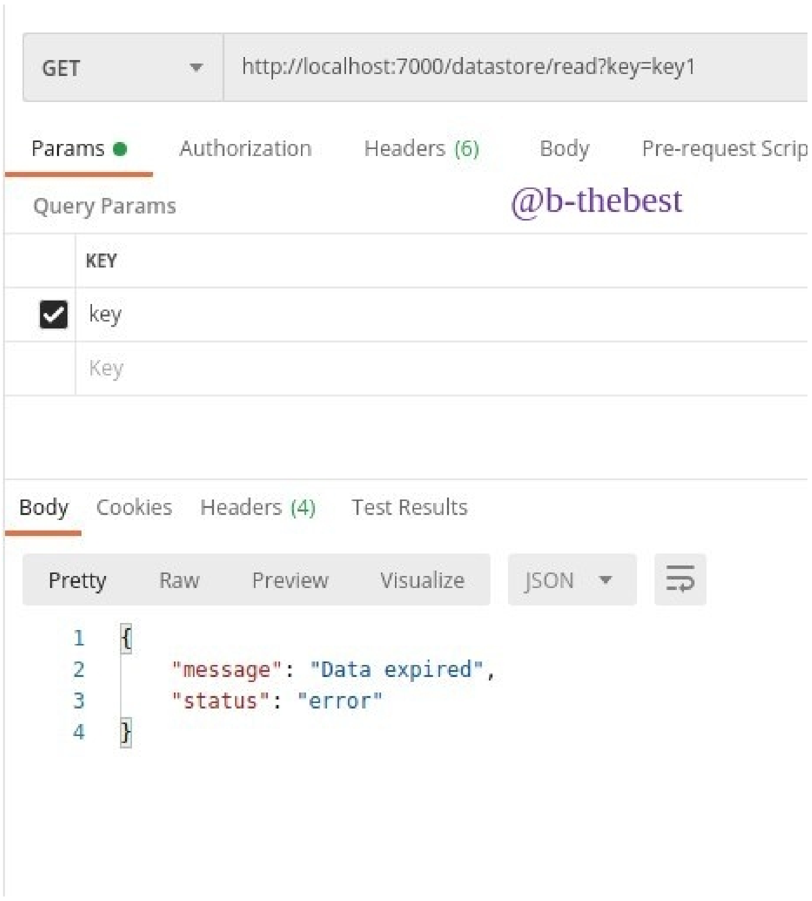
GET http://localhost:7000/datastore/read?key=key2

Query Params

KEY
✓ key
Key

```
{
    "message": "KeyError: No data for given key",
    "status": "error"
}
```

10. Trying to read key which is expired



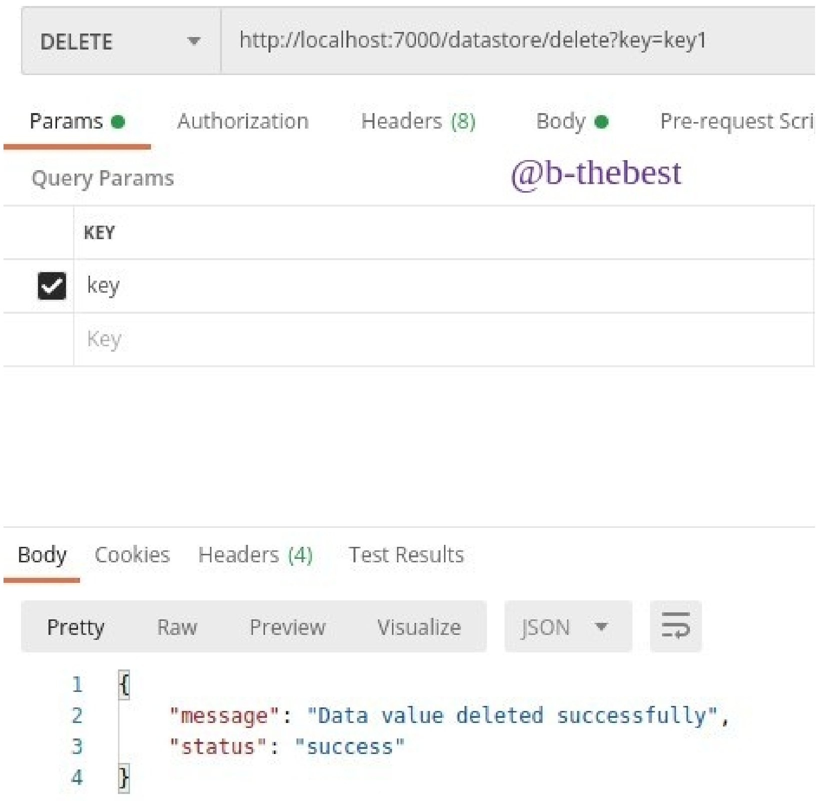
GET http://localhost:7000/datastore/read?key=key1

Query Params

KEY
✓ key
Key

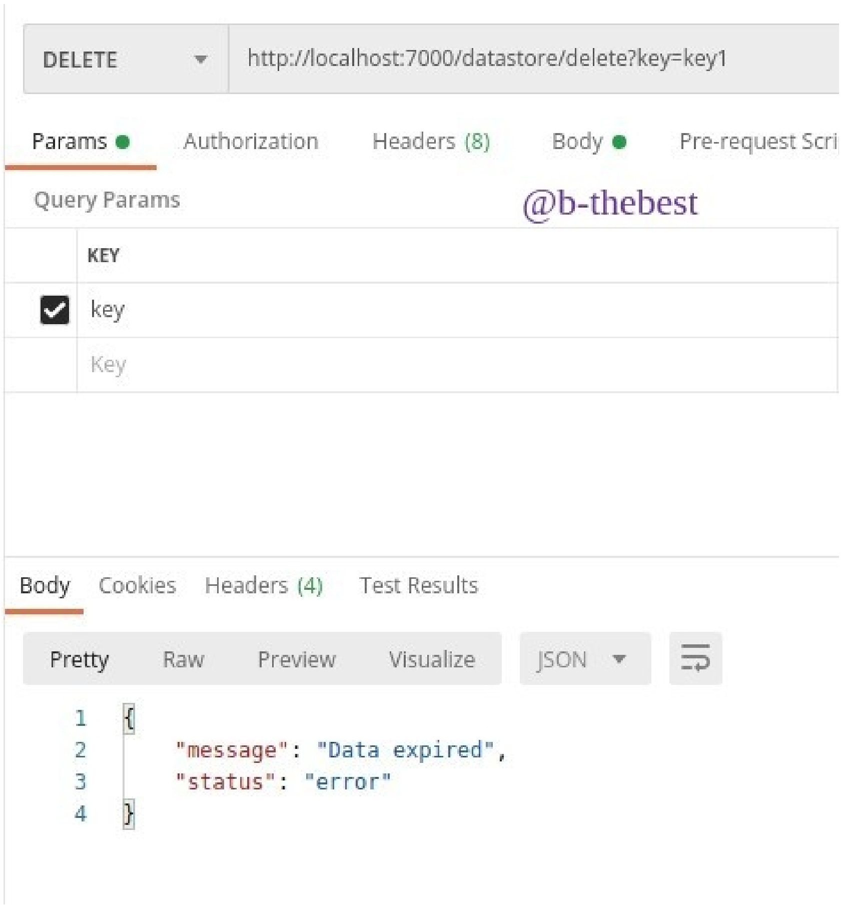
```
{
    "message": "Data expired",
    "status": "error"
}
```

11. Deleting some data from database



```
{
    "message": "Data value deleted successfully",
    "status": "success"
}
```

12. Trying to delete expired data from database



```
{
    "message": "Data expired",
    "status": "error"
}
```